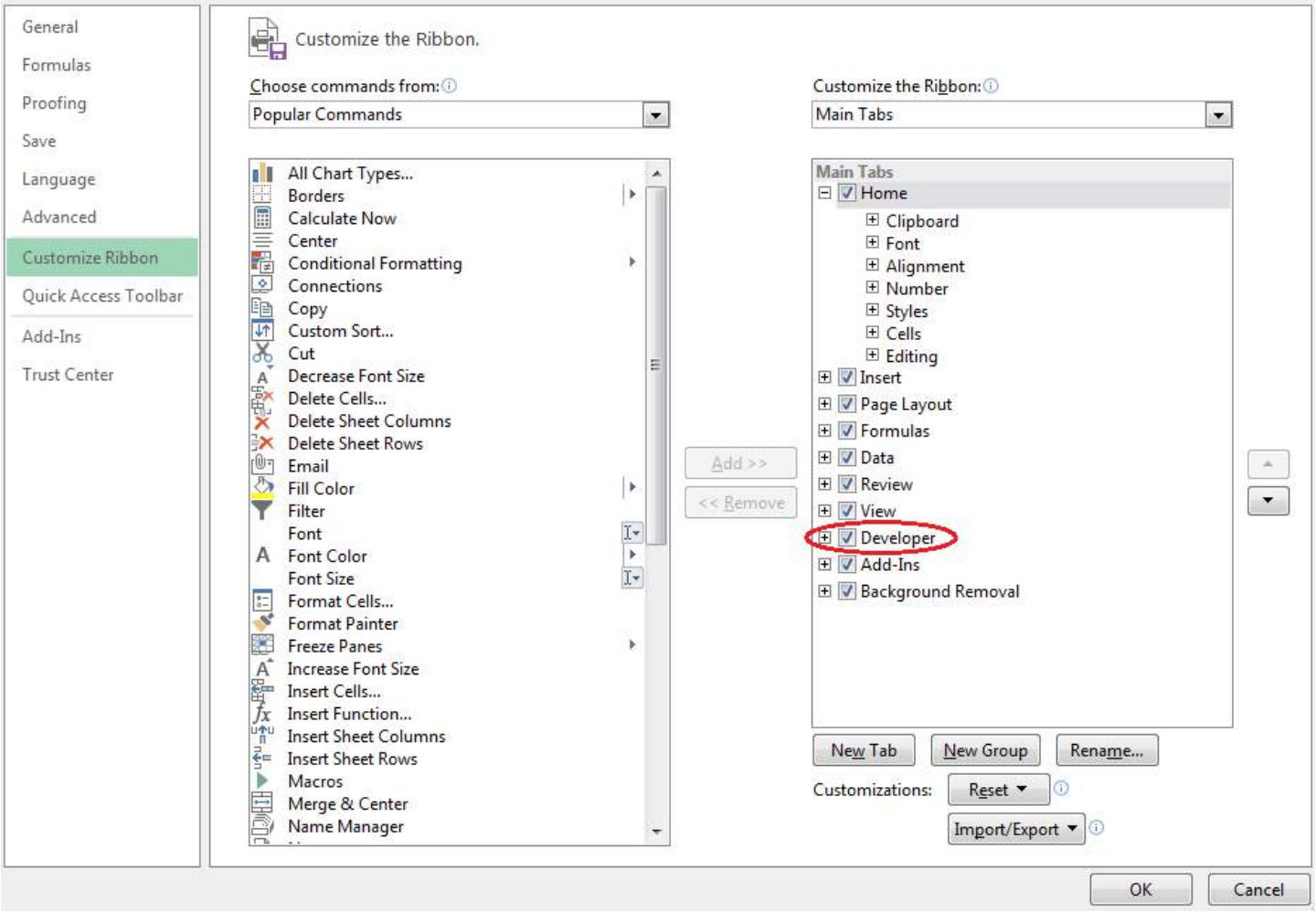
Chapter 1: Getting started with VBA

Version	Office Versions	Release Date Notes	Release Date
Vba6	? - 2007	[Sometime after][1]	1992-06-30
Vba7	2010 - 2016	[blog.techkit.com][2]	2010-04-15
VBA for Mac	2004, 2011 - 2016		2004-05-11

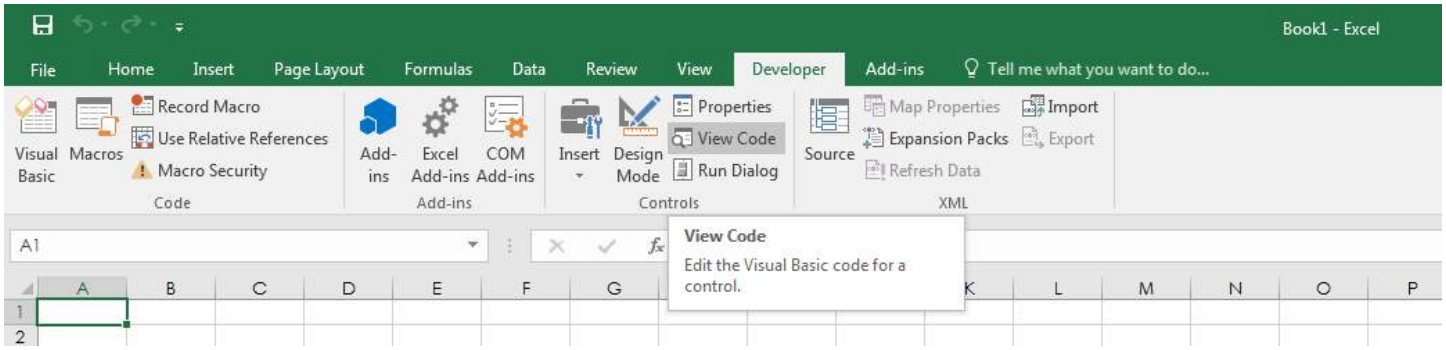
Section 1.1: Accessing the Visual Basic Editor in Microsoft Office

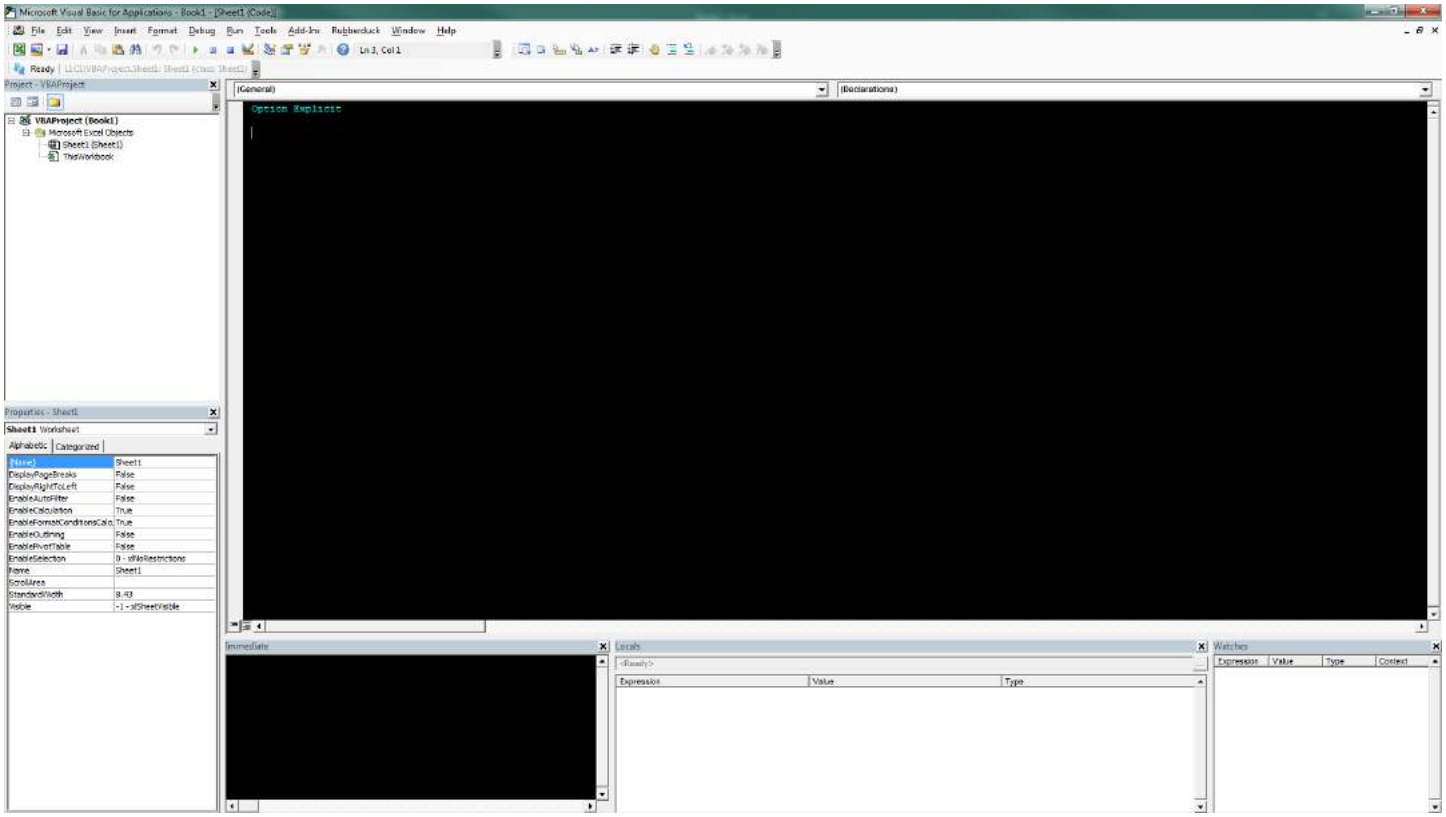
You can open the VB editor in any of the Microsoft Office applications by pressing `Alt` + `F11` or going to the Developer tab and clicking on the "Visual Basic" button. If you don't see the Developer tab in the Ribbon, check if this is enabled.

By default the Developer tab is disabled. To enable the Developer tab go to File -> Options, select Customize Ribbon in the list on the left. In the right "Customize the Ribbon" treeview find the Developer tree item and set the check for the Developer checkbox to checked. Click Ok to close the Options dialog.



The Developer tab is now visible in the Ribbon on which you can click on "Visual Basic" to open the Visual Basic Editor. Alternatively you can click on "View Code" to directly view the code pane of the currently active element, e.g. WorkSheet, Chart, Shape.





You can use VBA to automate almost any action that can be performed interactively (manually) and also provide functionality that is not available in Microsoft Office. VBA can create a document, add text to it, format it, edit it, and save it, all without human intervention.

Section 1.2: Debugging

Debugging is a very powerful way to have a closer look and fix incorrectly working (or non working) code.

Run code step by step

First thing you need to do during debugging is to stop the code at specific locations and then run it line by line to see whether that happens what's expected.

- Breakpoint (`F9`, Debug - Toggle breakpoint): You can add a breakpoint to any executed line (e.g. not to declarations), when execution reaches that point it stops, and gives control to user.
- You can also add the `Stop` keyword to a blank line to have the code stop at that location on runtime. This is useful if, for example, before declaration lines to which you can't add a breakpoint with `F9`
- Step into (`F8`, Debug - Step into): executes only one line of code, if that's a call of a user defined sub / function, then that's executed line by line.
- Step over (`Shift`+`F8`, Debug - Step over): executes one line of code, doesn't enter user defined subs / functions.
- Step out (`Ctrl`+`Shift`+`F8`, Debug - Step out): Exit current sub / function (run code until its end).

- Run to cursor (`Ctrl`+`F8`, Debug - Run to cursor): run code until reaching the line with the cursor.
- You can use `Debug.Print` to print lines to the Immediate Window at runtime. You may also use `Debug.?` as a shortcut for `Debug.Print`

Watches window

Running code line by line is only the first step, we need to know more details and one tool for that is the watch window (View - Watch window), here you can see values of defined expressions. To add a variable to the watch window, either:

- Right-click on it then select "Add watch".
- Right-click in watch window, select "Add watch".
- Go to Debug - Add watch.

When you add a new expression you can choose whether you just want to see it's value, or also break code execution when it's true or when its value changes.

Immediate Window

The immediate window allows you to execute arbitrary code or print items by preceeding them with either the `Print` keyword or a single question mark "`?`"

Some examples:

- `? ActiveSheet.Name` - returns name of the active sheet
- `Print ActiveSheet.Name` - returns the name of the active sheet
- `? foo` - returns the value of `foo`*
- `x = 10` sets `x` to 10*

* Getting/Setting values for variables via the Immediate Window can only be done during runtime

Debugging best practices

Whenever your code doesn't work as expected first thing you should do is to read it again carefully, looking for mistakes.

If that doesn't help, then start debugging it; for short procedures it can be efficient to just execute it line by line, for longer ones you probably need to set breakpoints or breaks on watched expressions, the goal here is to find the line not working as expected.

Once you have the line which gives the incorrect result, but the reason is not yet clear, try to simplify expressions, or replace variables with constants, that can help understanding whether variables' value are wrong.

If you still can't solve it, and ask for help:

- Include as small part of your code as possible for understanding of your problem
- If the problem is not related to the value of variables, then replace them by constants. (so, instead of `Sheets(a*b*c+d^2).Range(addressOfRange)` write `Sheets(4).Range("A2")`)
- Describe which line gives the wrong behaviour, and what it is (error, wrong result...)

Section 1.3: First Module and Hello World

To start coding in the first place, you have to right click your VBA Project in the left list and add a new Module. Your first *Hello-World* Code could look like this:

```
Sub HelloWorld()
    MsgBox "Hello, World!"
End Sub
```

To test it, hit the *Play*-Button in your Toolbar or simply hit the F5 key. Congratulations! You've built your first own VBA Module.